# Test Task

Write sql request, You have 2 tables:

"1-st table name - bet_win_sum_by_day. bet_win_sum_by_day have fields system_id, game_id, currency, date, bet_sum, win_sum. It's grouped by system_id, game_id, currency, date."

"2-d table name - events. Events tables is base source of data (not grouped for any field). Events table have fields date, time, user_id, system_id, currency. "

"Your task is to write sql request to get as result table with fields month_date, system_id, game_id, bet_sum, win_sum, user_count for August and September 2022."

Let's create a table for convenience and set the format to INT

```
8  use analyst_junior_test_task;
9  CREATE TABLE bet_win_sum_by_day (
10     system_id INT,
11     game_id INT,
12     currency INT,
13     date INT,
14     bet_sum INT,
15     win_sum INT
16 );
17
18 CREATE TABLE events (
19     date INT,
20     time INT,
21     user_id INT,
22     system_id INT,
23     currency INT
24 );
```

Write sql request to get as result:

```
select   bet_win_sum_by_day.system_id,
bet_win_sum_by_day.game_id,
bet_win_sum_by_day.bet_sum,
bet_win_sum_by_day.win_sum ,
MONTH(bet_win_sum_by_day.date),
count(events.user_id)
from  bet_win_sum_by_day

join events on bet_win_sum_by_day.system_id  = events.system_id
WHERE bet_win_sum_by_day.date between '2022-08-01' AND  '2022-09-30'

GROUP BY
MONTH(bet_win_sum_by_day.date),
 bet_win_sum_by_day.system_id,
 bet_win_sum_by_day.game_id
```

**Each of your answer needs visualisation. And 3-d task needs an explanation.**

Your tasks are:

1) To calculate average bet for each system and each game in it.

```
use analyst_junior_test_task;

select system_id, game_id, avg(bet_sum)  from data_base

group by system_id , game_id ;
```

Result Grid | Filter Rows:

| system_id | game_id | avg(bet_sum) |
|---|---|---|
| 1 | 100 | 120.9333 |
| 1 | 101 | 145.9333 |
| 1 | 102 | 110.5667 |
| 2 | 100 | 157.0588 |
| 2 | 101 | 168.5667 |
| 2 | 102 | 156.8333 |

2) To find game and system with biggest bet sum and avg bet.

```
select system_id ,  game_id,  avg(bet_sum)  from data_base

group by system_id , game_id

order by avg(bet_sum) desc
limit  1
```

Result Grid | Filter Rows:

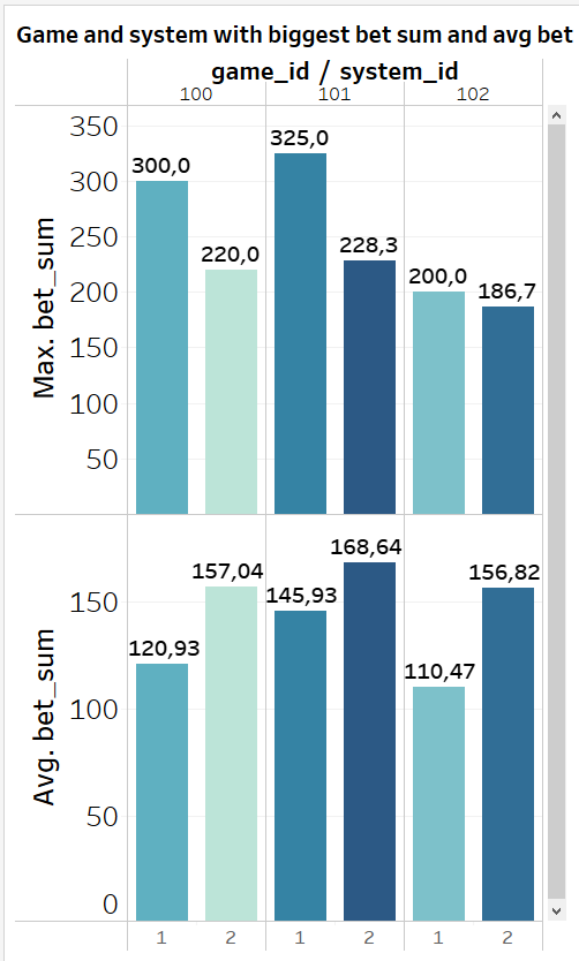
| system_id | game_id | avg(bet_sum) |
|---|---|---|
| 2 | 101 | 168.5667 |

```
17  select system_id ,  game_id,  max(bet_sum) from data_base
18
19  group by system_id , game_id
20
21  order by max(bet_sum)  desc
22
23  limit  1
24
25
26
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

| system_id | game_id | max(bet_sum) |
|---|---|---|
| 1 | 101 | 325 |

Now all results can be visualized using Tableau:

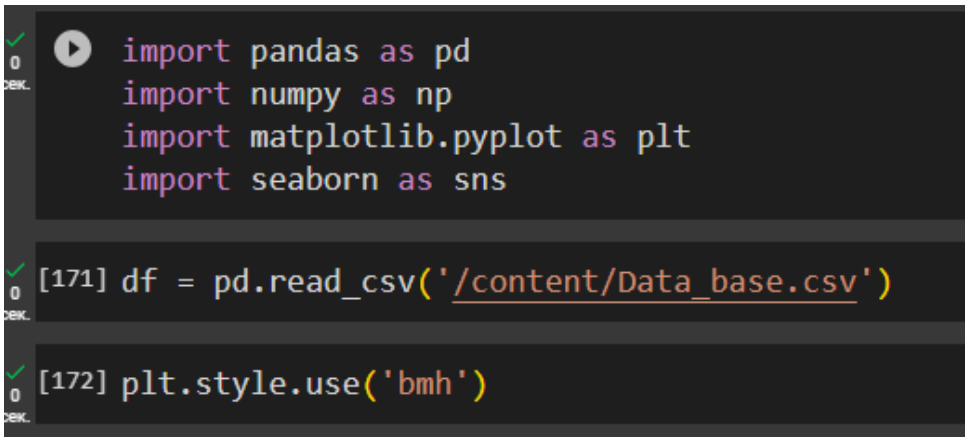


3) To find anomal activity in data.

**Legend**

| **field** | **description** |
|---|---|
| system_id | id of our partner |
| game_id | id of our game |
| date | date of data row |
| month_date | first date of month |
| currency | currency of operation in game |
| bet_sum | sum of bet in game |
| win_sum | sum of win in game |
| ggr | difference bet_sum - win_sum |
| round/spin | game round with some bet and win amount |
| round_count | number of rounds |
| avg_bet | average bet on 1 round |
| user_id | id of user |

Upload the data set to Colab



Check the data format

```
df
```

| | date | system_id | game_id | bet_sum | round_count |
|---|---|---|---|---|---|
| 0 | 2022-09-01 | 1 | 100 | 100.000000 | 10.0 |
| 1 | 2022-09-02 | 1 | 100 | 50.000000 | 15.0 |
| 2 | 2022-09-03 | 1 | 100 | 250.000000 | 23.0 |
| 3 | 2022-09-04 | 1 | 100 | 300.000000 | 14.0 |
| 4 | 2022-09-05 | 1 | 100 | 45.000000 | 10.0 |
| ... | ... | ... | ... | ... | ... |
| 162 | 2022-09-26 | 2 | 102 | 141.666667 | 20.0 |
| 163 | 2022-09-27 | 2 | 102 | 145.000000 | 20.0 |
| 164 | 2022-09-28 | 2 | 102 | 186.666667 | 24.5 |
| 165 | 2022-09-29 | 2 | 102 | 144.166667 | 21.0 |
| 166 | 2022-09-30 | 2 | 102 | 170.000000 | 21.0 |

167 rows × 5 columns

```
[126] df.dtypes
```

```
date            object
system_id        int64
game_id          int64
bet_sum        float64
round_count    float64
dtype: object
```

Converts date  to datetime object and  divide by days

```
[127] df['date'] = pd.to_datetime(df['date'])
```

```
df['day'] = df['date'].dt.day
```

```
df
```

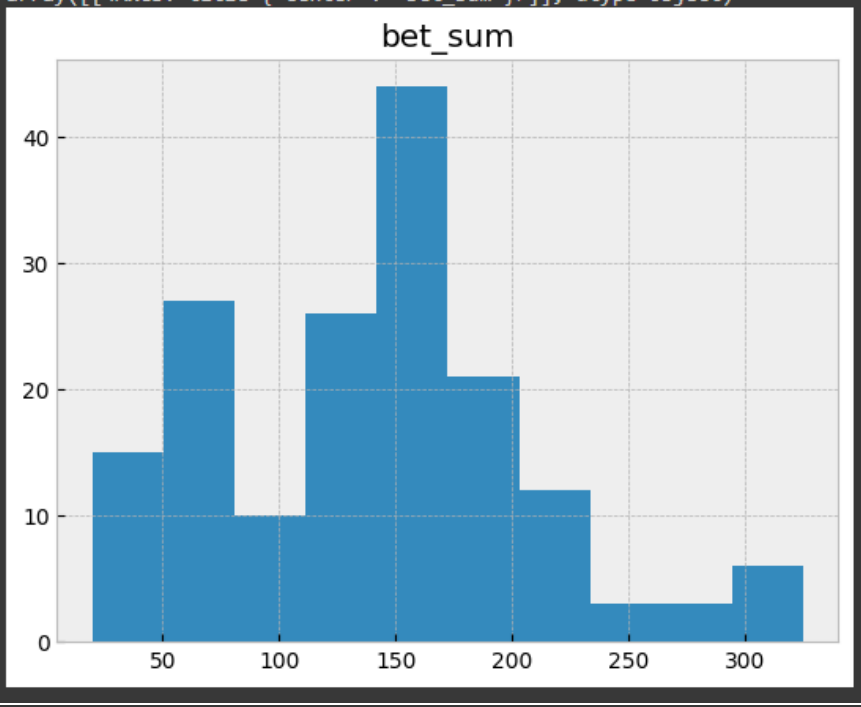
| | date | system_id | game_id | bet_sum | round_count | day |
|---|---|---|---|---|---|---|
| 0 | 2022-09-01 | 1 | 100 | 100.000000 | 10.0 | 1 |
| 1 | 2022-09-02 | 1 | 100 | 50.000000 | 15.0 | 2 |
| 2 | 2022-09-03 | 1 | 100 | 250.000000 | 23.0 | 3 |
| 3 | 2022-09-04 | 1 | 100 | 300.000000 | 14.0 | 4 |
| 4 | 2022-09-05 | 1 | 100 | 45.000000 | 10.0 | 5 |
| ... | ... | ... | ... | ... | ... | ... |
| 162 | 2022-09-26 | 2 | 102 | 141.666667 | 20.0 | 26 |
| 163 | 2022-09-27 | 2 | 102 | 145.000000 | 20.0 | 27 |
| 164 | 2022-09-28 | 2 | 102 | 186.666667 | 24.5 | 28 |
| 165 | 2022-09-29 | 2 | 102 | 144.166667 | 21.0 | 29 |
| 166 | 2022-09-30 | 2 | 102 | 170.000000 | 21.0 | 30 |

167 rows × 6 columns

Let's build a histogram by bet_sum and see if there is a normal distribution:  We see there are outliers (tail) from 50 to 100 and around 300 there is a tail:

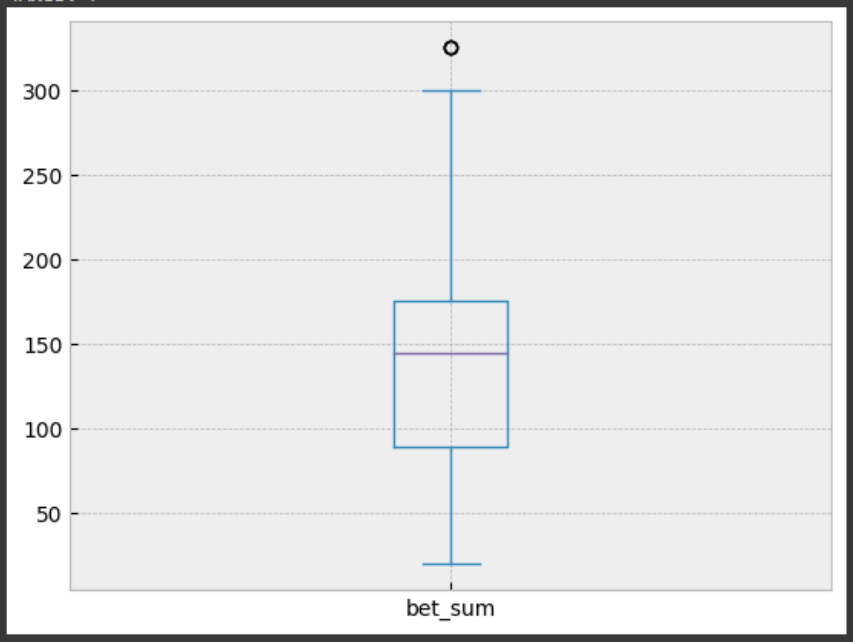
```
df.hist('bet_sum')
```

```
array([[<Axes: title={'center': 'bet_sum'}>]], dtype=object)
```




```
df['bet_sum'].plot(kind='box')
```

```
<Axes: >
```



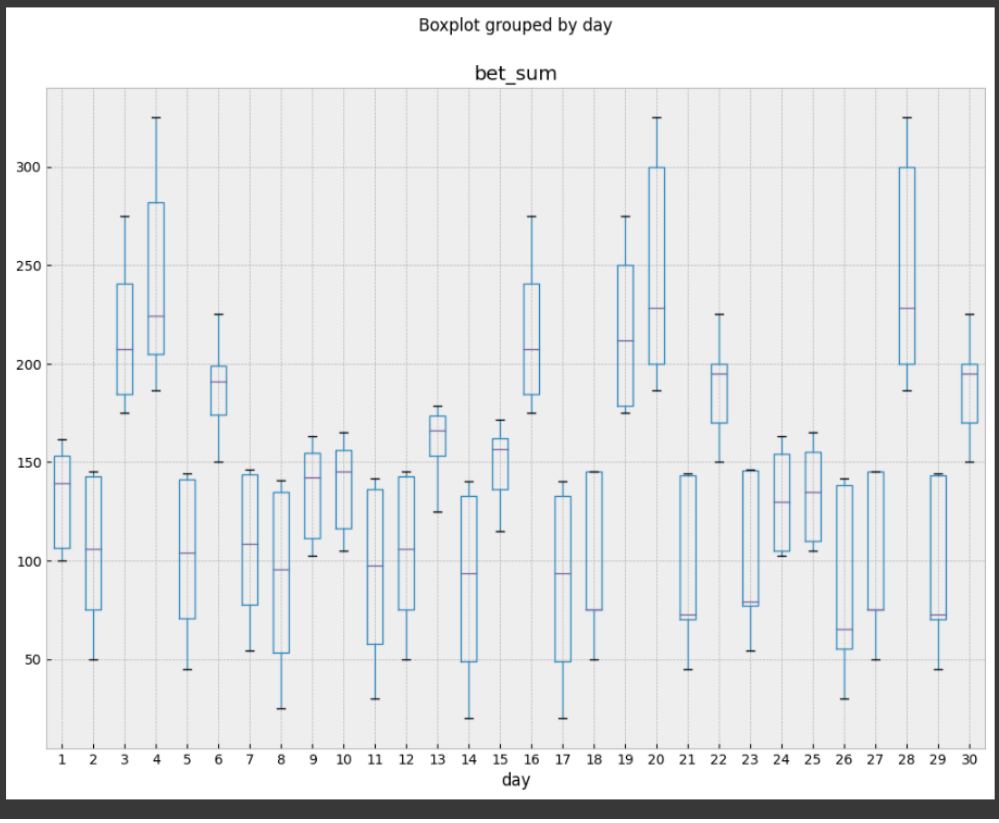
Let's divide the days into groups:
We see how the amount drops by the middle of the week, then rises again after the 20th and drops again by the end of the month.

```
df.groupby('day_group')['bet_sum'].agg('median')
```

```
day_group
(2022-08-31 23:59:59.999999999, 2022-09-07 12:00:00]    148.166667
(2022-09-07 12:00:00, 2022-09-14]                       132.500000
(2022-09-14, 2022-09-22]                                163.333333
(2022-09-22, 2022-09-30]                                142.500000
Name: bet_sum, dtype: float64
```

We see these outliers on the graph.We see these same outliers on the graph,later we will check whether these are really anomalies using the three sigma rule.

```
df.boxplot(column='bet_sum', by='day', figsize=(12,9))
plt.show()
```



Let's see if there is a connection between the variables:
We see a linear connection between round_cound and bet_sum , but this is a linear connection and it is better not to use it for intelligence data.

```
df.corr()
```

```
<ipython-input-138-2f6f6606aa2c>:1: FutureWarning: The default value of numeric_only in D
  df.corr()
```

| | system_id | game_id | bet_sum | round_count | day |
|---|---|---|---|---|---|
| system_id | 1.000000 | 0.105641 | 0.272960 | -0.006390 | -0.083386 |
| game_id | 0.105641 | 1.000000 | -0.014768 | -0.112569 | 0.104352 |
| bet_sum | 0.272960 | -0.014768 | 1.000000 | 0.440125 | 0.001942 |
| round_count | -0.006390 | -0.112569 | 0.440125 | 1.000000 | 0.389473 |
| day | -0.083386 | 0.104352 | 0.001942 | 0.389473 | 1.000000 |

Let's try to find connections through Phik:

```
import phik
from phik.report import plot_correlation_matrix
from phik import report
```

```
[141] phik_overview = df.phik_matrix()
```

```
interval columns not set, guessing: ['system_id', 'game_id', 'bet_sum', 'round_count', 'day']
```

```
[143] phik_overview
```

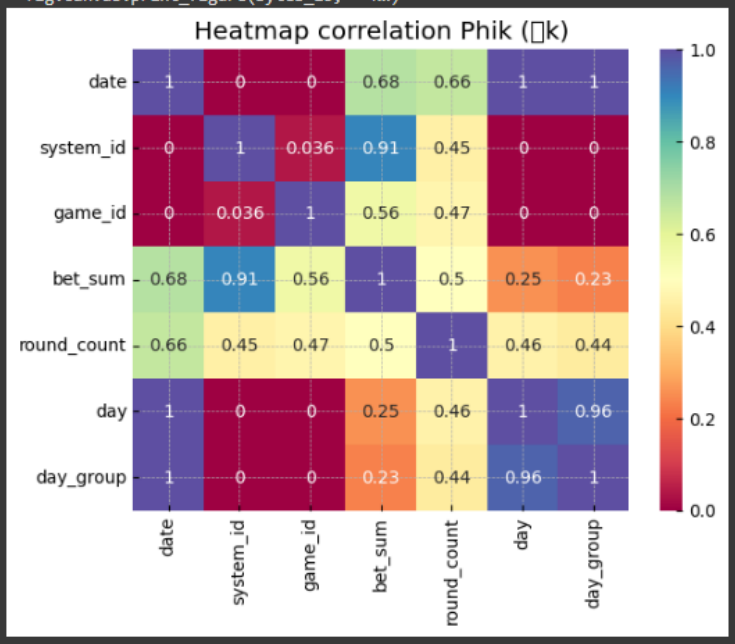
| | date | system_id | game_id | bet_sum | round_count | day | day_group |
|---|---|---|---|---|---|---|---|
| date | 1.000000 | 0.000000 | 0.000000 | 0.680418 | 0.659768 | 1.000000 | 1.000000 |
| system_id | 0.000000 | 1.000000 | 0.036308 | 0.906103 | 0.449808 | 0.000000 | 0.000000 |
| game_id | 0.000000 | 0.036308 | 1.000000 | 0.560737 | 0.467578 | 0.000000 | 0.000000 |
| bet_sum | 0.680418 | 0.906103 | 0.560737 | 1.000000 | 0.502458 | 0.254085 | 0.231612 |
| round_count | 0.659768 | 0.449808 | 0.467578 | 0.502458 | 1.000000 | 0.457792 | 0.440890 |
| day | 1.000000 | 0.000000 | 0.000000 | 0.254085 | 0.457792 | 1.000000 | 0.963149 |
| day_group | 1.000000 | 0.000000 | 0.000000 | 0.231612 | 0.440890 | 0.963149 | 1.000000 |

Let's build a heatmap for viewing communications:

```
[144] sns.heatmap(phik_overview, cmap='Spectral', annot=True)
      plt.title('Heatmap correlation Phik (ϕk)')
```

```
sns.heatmap(phik_overview, cmap='Spectral', annot=True)
plt.title('Heatmap correlation Phik (ϕk)')
```

```
Text(0.5, 1.0, 'Heatmap correlation Phik (ϕk)')/usr/local/lib/python3.10/dist-packages/IPyth
  func(*args, **kwargs)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Glyph 1
  fig.canvas.print_figure(bytes_io, **kw)
```



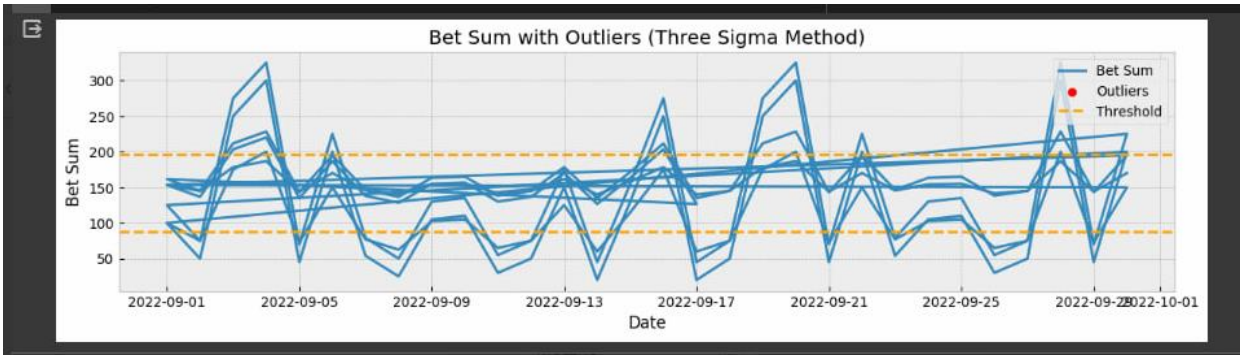

```
phik_overview['bet_sum'].sort_values(ascending=False)
```

```
bet_sum         1.000000
system_id       0.906103
date            0.680418
game_id         0.560737
round_count     0.502458
day             0.254085
day_group       0.231612
Name: bet_sum, dtype: float64
```

There is a relationship between the bet amount and the date of about 68 percent.

Now let's check whether the emissions were really anomalous using three sigma.

```python
def detect_outliers(df, bet_sum):
    mean = df[bet_sum].mean()
    std = df[bet_sum].std()
    threshold = 3 * std

    outliers = df[(df[bet_sum] > mean + threshold ) | (df[bet_sum] < mean - threshold)]
    return outliers


outliers = detect_outliers(df, 'bet_sum')


plt.figure(figsize=(28, 10))

plt.subplot(2, 1, 1)
plt.plot(df['date'], df['bet_sum'], label='Bet sum')
plt.scatter(outliers['date'], outliers['bet_sum'],)
plt.axhline(mean + threshold, color='orange', linestyle='--', label='Threshold')
plt.axhline(mean - threshold, color='orange', linestyle='--')
plt.title('Bet Sum with Outliers (Three sigma method)')
plt.xlabel('Date')
plt.ylabel('Bet Sum')
plt.legend()

plt.tight_layout()
plt.show()
```




According to the rules of three sigma, outliers are indeed anomalous. We see how there are anomalies at the beginning of the month and in the middle, as well as at the end.